案例 6-椭圆中点 Bresenham 算法

文档编写：霍波魏

校稿/修订：孔令德

时间 2019~2020

联系方式：QQ997796978

说明：本套案例由孔令德开发，原版本为 Visual C++6.0，配套于孔令德的著作《计算机图形学-基于 MFC 三维图形开发》一书。孔令德计算机工程研究所的学生霍波魏在学习计算机图形学期间，对本套案例进行了升级并编写了学习文档。现在程序的编写和程序的解释都是基于 Windows 10 操作系统，使用 Microsoft visual studio 2017 平台的 MFC（英文版）开发。

一、知识点

1. CEllipse 类

设计画椭圆的 CEllipse 类，用四分法画椭圆。

2. 四分画椭圆法

椭圆是长半轴和短半轴不相等的圆，椭圆的扫描转换与圆的扫描转换有类似之处，通过顺时针绘制 1/4 椭圆的中点 Bresenham 算法原理，根据对称性可以绘制完整椭圆，具体的分析见图 6-1 和 6-2。

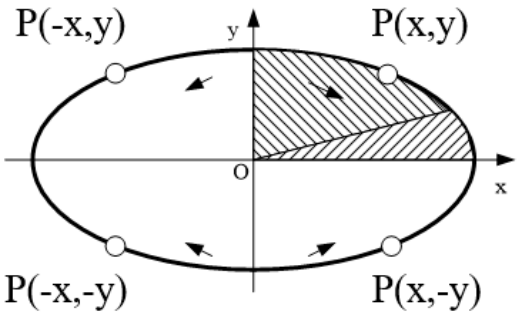

图 6-1 椭圆的对称性

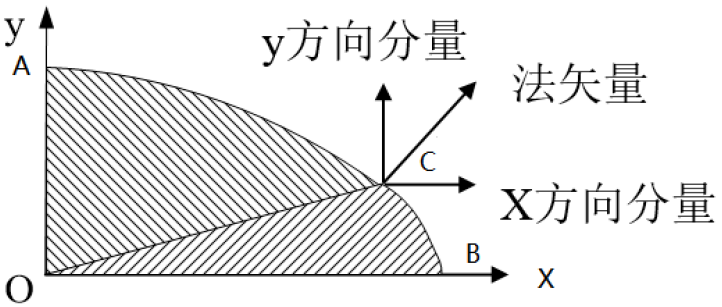

图 6-2 椭圆法矢量

由图 2 可以观察到：第一部分的 AC 椭圆弧段，法矢量的 x 向分量小于 y 向分量，斜率 k 处处满足|k|＜1，|Δx|＞|Δy|，所以 x 方向为主位移方向；在 C 点，法矢量的 x 向分量等于 y 向分量，斜率 k 满足 k＝－1，|Δx|＝|Δy|；第二部分的 CB 椭圆弧段，法矢量的 x 向分量大于 y 向分量，斜率 k 处处满足|k|＞1，|Δy|＞|Δx|，所以 y 方向为主位移方向。

二、案例描述

本案例通过确定长半轴和短半轴，绘制以坐标圆心为中心，通过短半轴的递增，使得同心椭圆由小变大，形成两个相互垂直的同心椭圆。

三、实现步骤

1. 在 CTestView 类中添加 DoubleBuffer()双缓冲函数。
2. 添加 CEllipse 类并在其中用四分法画椭圆。
3. 在 CTestView 类中添加 DrawObject 函数，并在其中进行同心椭圆的绘制。
4. 在 CTestView 类的构造函数中对变量进行初始化。
5. 在 OnDraw()中对 SetTimer()和 DoubleBuffer()进行调用。

四、主要算法

1. CEllipse 类

```
public:
    CEllipse();
    ~CEllipse();
    void SetData(double a, double b, CP2i p0);
    void Draw(CDC* pDC);//绘制椭圆
    void EllipsePoint(CP2i p, CDC* pDC, CRGB clr);//四分法画椭圆子函数
private:
    double a, b;//椭圆的长短半轴
    CP2i CenterPoint;//椭圆中心
    CRGB clr;
void CEllipse::SetData(double a, double b, CP2i p0)
{
    this->a = a;
    this->b = b;
    this->CenterPoint = p0;
}
void CEllipse::Draw(CDC* pDC)//绘制椭圆
{
```

```
        CP2i p;
        double d1, d2;
        p.x = 0;
        p.y = ROUND(b);
        d1 = b * b + a * a*(-b + 0.25);
        EllipsePoint(p, pDC, clr);
        while (b * b * (p.x + 1) < a * a * (p.y - 0.5))//椭圆 AC 弧段
        {
            if (d1 < 0)
                d1 += b * b*(2 * p.x + 3);
            else
            {
                d1 += b * b*(2 * p.x + 3) + a * a*(-2 * p.y + 2);
                p.y--;
            }
            p.x++;
            EllipsePoint(p, pDC, clr);
        }
        d2 = b * b*(p.x + 0.5)*(p.x + 0.5) + a * a*(p.y - 1)*(p.y - 1) - a * a *
b * double(b);//椭圆 CB 弧段
        while (p.y > 0)
        {
            if (d2 < 0)
            {
                d2 += b * b*(2 * p.x + 2) + a * a*(-2 * p.y + 3);
                p.x++;
            }
            else
                d2 += a * a*(-2 * p.y + 3);
            p.y--;
            EllipsePoint(p, pDC, clr);
        }
    }
    void CEllipse::EllipsePoint(CP2i p, CDC* pDC, CRGB clr)//四分法画椭圆子函数
    {
        COLORREF c = CRGBTORGB(clr);
        pDC->SetPixelV(p.x + CenterPoint.x, p.y + CenterPoint.y, c);
        pDC->SetPixelV(-p.x + CenterPoint.x, p.y + CenterPoint.y, c);
        pDC->SetPixelV(p.x + CenterPoint.x, -p.y + CenterPoint.y, c);
        pDC->SetPixelV(-p.x + CenterPoint.x, -p.y + CenterPoint.y, c);
    }
```

2.CTestView 类

```
    public:
        void DoubleBuffer(CDC* pDC);//双缓冲
```

```
        void DrawObject(CDC* pDC);//绘制图形
    protected:
        BOOL bPlay;
        CEllipse ellipse[60];
        int nA, nB;//椭圆的长短半轴
    // CTestView message handlers
    void CTestView::DoubleBuffer(CDC*pDC)
    {
        CRect rect;//定义客户区矩形
        GetClientRect(&rect);//获得客户区的大小
        pDC->SetMapMode(MM_ANISOTROPIC);//pDC 自定义坐标系
        pDC->SetWindowExt(rect.Width(), rect.Height());//设置窗口范围
        pDC->SetViewportExt(rect.Width(), -rect.Height());
                                //设置视区范围,x 轴水平向右，y 轴垂直向上
        pDC->SetViewportOrg(rect.Width() / 2, rect.Height() / 2);
                                //客户区中心为原点
        CDC memDC;//内存 DC
        CBitmap NewBitmap, *pOldBitmap;//内存中承载的临时位图
        memDC.CreateCompatibleDC(pDC);//创建一个与显示 pDC 兼容的内存 memDC
        NewBitmap.CreateCompatibleBitmap(pDC, rect.Width(), rect.Height());
                                //创建兼容位图
        pOldBitmap = memDC.SelectObject(&NewBitmap);//将兼容位图选入 memDC
        memDC.SetMapMode(MM_ANISOTROPIC);//memDC 自定义坐标系
        memDC.SetWindowExt(rect.Width(), rect.Height());
        memDC.SetViewportExt(rect.Width(), -rect.Height());
        memDC.SetViewportOrg(rect.Width() / 2, rect.Height() / 2);
        rect.OffsetRect(-rect.Width() / 2, -rect.Height() / 2);
        DrawObject(&memDC);//向 memDC 绘制图形
        pDC->BitBlt(rect.left, rect.top, rect.Width(), rect.Height(), &memDC,
-rect.Width() / 2, -rect.Height() / 2, SRCCOPY);//将内存 memDC 中的位图拷贝到显示 pDC
中
        memDC.SelectObject(pOldBitmap);//恢复位图
        NewBitmap.DeleteObject();//删除位图
    }
    void CTestView::DrawObject(CDC* pDC)
    {
        for (int i = 0; i < 60; i++)
        {
            if (0 == i % 2)
            {
                ellipse[i].SetData(nA + i * 20, nB + i * 10, CP2i(0, 0));
                ellipse[i].Draw(pDC);
            }
            else
```

```
            {
                ellipse[i].SetData(nB + i * 10, nA + i * 20, CP2i(0, 0));
                ellipse[i].Draw(pDC);
            }
        }
    }
```

五、实现效果

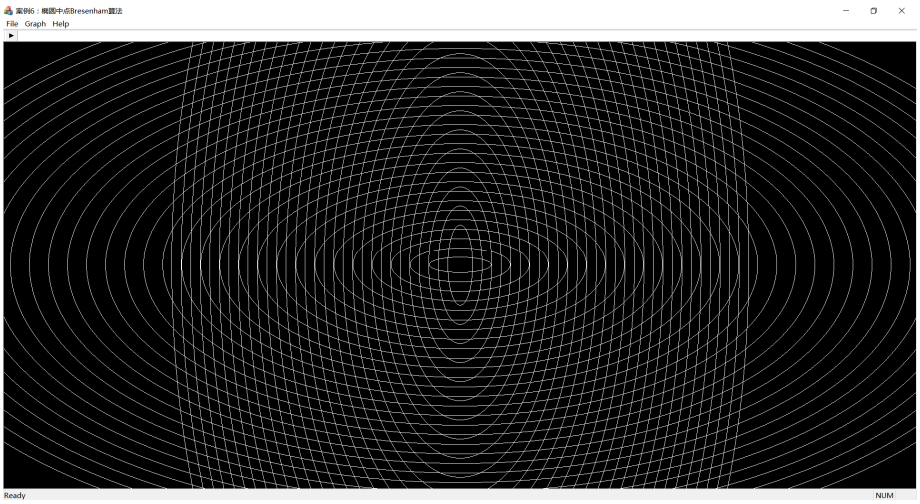

图 6-3 椭圆中点 Bresenham 算法效果图

六、案例心得

将本案例与案例五进行对比，相同点是椭圆和圆都满足对称性，区别是：一个圆可以等分为 8 份，只要绘制出其中的一份，根据对称性就可以绘制出整个圆；而椭圆则不同，椭圆只可以等分成四分，以第一象限为例，1/4 椭圆在第一象限由于法矢量在 X 轴和 Y 轴的分量的改变可以分为两部分，第一部分是法矢量的 x 向分量小于 y 向分量，斜率 k 处处满足$|k|<1$，$|\Delta x|>|\Delta y|$，所以 x 方向为主位移方向，第二部分是法矢量的 x 向分量大于 y 向分量，斜率 k 处处满足$|k|>1$，$|\Delta y|>|\Delta x|$，所以 y 方向为主位移方向，所以在绘制椭圆时只可以将其分成四份，再通过对称性绘制完整椭圆。